Review of C++ Workshop Solutions

Copyright © James Raynard 2021

URL Container

- Create a class to hold an ordered history of URLs (as might be used in a web browser). Use an STL container to store the URLs
- Ensure that a URL is added to the history only if it is not in the history already. If it is, it should be moved to the beginning of the history
- Write a program to test your class

Copyright © James Raynard 2021

String input

- Write a program that reads strings entered by the user, stores them in a vector and prints out the strings in the order they were entered
- Change your program to use a list instead of a vector
- Change your program to use a deque instead of a list
- Change the list and deque versions to store the strings in reverse order, without performing any manipulation on their elements

Copyright © James Raynard 2021

Maps

- Write a program that reads words from input
- Store the value and length of each word in a suitable std::pair
- Store each pair in
  - A vector
  - A map whose key and value have the appropriate types
- Print out all the elements of the vector and the map

Copyright © James Raynard 2021

Overloading and override

- What is meant by
  - Overloading a member function?
  - To overload a member function, we add another function in the same class that has the same name but takes different arguments
  - Overriding a member function?
  - The member function that we wish to override must be declared virtual in the parent class
  - To override this member function, we add a function in a child class that has the same name and takes the same arguments

Copyright © James Raynard 2021

Inheritance

- Write a hierarchy of three classes, Base, Child and Grandchild in which Child inherits from Base and Grandchild inherits from Child
- Give each class a print() member function which prints out the name of the class
- Write a print_class() function which takes a Base argument by value
- Create an instance of each class and call print_class()
- Explain your results
  - In all three cases, the name of the Base class is printed out
  - When an object is passed by value, its static type is always used
  - The member function will be called on the Base class

Copyright © James Raynard 2021

Inheritance 2

- Change the print() member function to be virtual
- Run the program again and explain your results
  - In all three cases, the name of the Base class is printed out
  - When an object is passed by value, its static type is always used
  - The member function will be called on the Base class
- Change the print_function() argument to be passed by reference
  - The correct name of the class is printed out
  - When an object is passed by reference or by address, its dynamic type is used
  - A virtual function will be called on the dynamic type of the object - Base, Child or Grandchild

Copyright © James Raynard 2021

Inheritance 3

- Add an integer member to the Base class and a constructor which initializes it from its argument
- Why does the program not compile?
  - The compiler synthesizes a default constructor for Child which calls the default constructor for Base
  - However, since Base defines a non-default constructor, the compiler does not synthesize a default constructor for it
  - When main() creates a Child object, it calls the Child default constructor, which calls the Base default constructor, which is not defined
  - Hence there is a compiler error

Copyright © James Raynard 2021

Inheritance 3

- What is the best way to solve this problem?
  - We could add a default constructor to Base, or implement a constructor for Child and Grandchild which takes an int
  - If we add a default constructor for Base, we need to choose a suitable value for the int member (by default, it will have an undefined value)
  - A constructor for Child and Grandchild would correctly initialize the int member from a value supplied by its caller
  - It would also give the hierarchy a consistent interface
  - This is the preferred solution

Copyright © James Raynard 2021

Buffer Manager class

- The buffer manager class is a simple class that manages a resource (in this case, a memory buffer)
- It has the following interface:

```
class BufferManager {
   private:
      int size;
      char *buffer;
   public:
   BufferManager(int size=10);
   ~BufferManager();
   BufferManager(const BufferManager &other);
   BufferManager& operator =(const BufferManager& other);
};
```

Copyright © James Raynard 2021

Buffer Manager class

- The Buffer Manager dynamically allocates the memory buffer in its constructor as an array of char
  - The constructor argument gives the number of bytes in the array
- The copy constructor and assignment operator perform a deep copy of the array
- The Buffer Manager's destructor releases the memory buffer
- What is the name for this idiom?
  - RAII - Release Acquisition Is Initialization
- Implement the Buffer Manager in its own namespace

Copyright © James Raynard 2021

Buffer Manager class

- Write a program to test the class
  - This should exercise the copy constructor and assignment operator
  - It should also have a suitable handler for any exceptions that might be thrown
- Check that your program runs as expected in the absence of exceptions

Copyright © James Raynard 2021

Buffer Manager Constructor

- Alter your program so that the constructor throws an std::bad_alloc exception
- Check that the exception is handled correctly
- Is the class destructor called? Explain your answer
  - In C++, an object does not exist until the constructor has successfully completed its execution
  - If the constructor throws, no object exists, and there is nothing to destroy. The destructor is not called
- Add print statements to the program and check the results

Copyright © James Raynard 2021

Buffer Manager Copy Constructor

- Alter your program so that the copy constructor throws an std::bad_alloc exception
- Check that the exception is handled correctly
- Is the class destructor called? Explain your answer
  - The destructor will not be called for the constructed object, because it does not exist
  - The destructor will be called for the copied object and any other objects that exist in the scope
- Add print statements to the program and check the results

Copyright © James Raynard 2021

Buffer Manager Assignment Operator

- Alter your program so that the assignment operator throws an std::bad_alloc exception
- Check that the exception is handled correctly
- Is the class destructor called? Explain your answer
  - The destructor will be called for all objects that exist in the scope, including the ones involved in the assignment
- Add print statements to the program and check the results

Copyright © James Raynard 2021